Real-time Operating Systems and Systems Programming

Security

ARVUTITEADUSE INSTITUUT
DEPARTMENT OF COMPUTER SCIENCE

Security Topics

- General issues
- Specific issues
- C specific issues

But First!

- http://www.youtube.com/watch?feature=player_embedded&v=p5T81yHkHtI

Sources

- http://www.dwheeler.com/secure-programs/
- http://www.ibm.com/developerworks/library/s-buffer-defend.html
- https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Coding+Standard

Posssible problems

- Disruption of work
- Data integrity
- Privilege escalation
- Data leakage

(CIA triad: confidentiality, integrity, availability)

What Causes Security Issues?

- Lack of skills
- Insecure tools (C language)
- Multi-user and parallel processes are difficult to predict and think about
- Lazyness
- Time/Money
- There is lack of good programmers
- User is not interested

... ?

Paranoia

- Being paranoid is the foundation of security
- Think like an attacker
  bodyguard analogy
- Presume that the attacker has the ability to exploit any weaknesses
- The defender must always be defensive, the attacker only needs one successful attack
  First World War counter analogy here

Guard your inputs

- Input is a lie!

Command line

- Execve() lets you add \0 chars where not expected
- Setuid/setgid problems

Environment variables

- You have full control over environment
- IFS variable (telling what character separates the commands in a shell)
- When you use system() function, causes problems
- Solution: purify the environment; use only what needed
- (setuid/setgid problems)
- User gets to include random .so files using LD_PRELOAD (and change it in ~/.environment variable)

Filenames

- Sneaky . .. and / possibilities
- Buffer overrun with PATH_MAX problems
- ../*/../*/../*/../* denial of service when using glob() function

Passwords

- Problem: how to ask password so that it does not reach the screen of the user.
- "Solution":

```
#include <unistd.h>
char * = getpasswd(char * prompt)
```

- Connects to "real" terminal /dev/tty , if cannot, tries stdin ja stderr . Blocks INTR, QUIT and SUSP commands in terminal.
- Terminal is flushed before and after password is typed

What does getpass() do?

- Prints the prompt
- Goes into noncanonical mode, turns off echo, restores the terminal state after function
- Due to lack of *thread-safety*, and exclusion from POSIX standard, general recommendation not to use it. Write yourself (or find a working solution).
- For important applications the good practice is to encrypt the password upon recieval and overwrite the original buffer.

Encryption: crypt()

- Encrypts using DES (broken) or MD5 (broken soon); Blowfish or SHA-256 / SHA-512:

```
char * crypt(constchar* key, const char* salt);
```

- Belief that hash, once calculated cannot be reversed in a reasonable time.
- salt: if two letters, chooses DES, if MD5, start the string as 1 + 8 chars, which end in $ or \0
- For Blowfish etc see manpage; you change the id

Salt

- Salting prevents dictionary attacks using rainbow tables.
- Output being salt + $ (when missing) + hash
- Salt should be a random string when the password is stored
- For checking the password, provide the previous output of crypt() as the salt, and compare *salt* to crypt() result. (As $ ends salt, you can provide the whole result for salt argument)

Storage of passwords

- Hash is problematic; MD5 has over 9000 million tries per second
- You can calculate the hash repetitively on existing hashes: try it 100 times to send attacker away
- The attacker using a GPU will be thwarted
- Don't invent stuff, use the bcrypt library

Stack smashing

- Canary
  - Ubuntus uses by default, others not
- Address space randomization (ASLR)

Standard library problems

- Mostly the lack of input length checks

Malloc

- Double free() really problematic
- You can control the behaviour by setting MALLOC_CHECK 2 environment variable
- After the release, use a macro to set the pointer to NULL

Non-negative values

- Use an unsigned type

Compilation suggestion

- gcc -Wall -Wpointer-arith -Wstrict-prototypes -O2

Be paranoid!